Table of Contents

Underapproximative verification of an automated anesthesia delivery system

This example will demonstrate the use of https://abyvinod.github.io/SReachTools/SReachTools in verification and controller synthesis for stochastic continuous-state discrete-time linear time-invariant (LTI) systems.

In this example, we will verify an automated anesthesia delivery model.

We wish to ascertain the set of intial states (patient sedation levels) from which the automated anesthesia delivery system can continue to maintain within pre-specified safe bounds.

```
% Prescript running
close all;clc;clear;
srtinit
```

Problem Formulation

We first define a `LtiSystem` object corresponding to the discrete-time approximation of the three-compartment pharmacokinetic system model.

We bound the anesthesia the automation can deliver to $[0, 7]$ mg/dL and account for patient model mismatch via an additive Gaussian noise.

```
% System matrices: State matrix and input matrix
% ------------------------------------------------
systemMatrix = [0.8192, 0.03412, 0.01265;
                0.01646, 0.9822, 0.0001;
                0.0009, 0.00002, 0.9989];
inputMatrix = [0.01883;
               0.0002;
               0.00001];
% Input bounds
% ------------
auto_input_max = 7;

% Process disturbance with a specified mean and variance
% -----------------------------------------------------
```

```
dist_mean = 0;
dist_var = 5;
process_disturbance = RandomVector('Gaussian',dist_mean, dist_var);

% LtiSystem definition
sys = LtiSystem('StateMatrix', systemMatrix, ...
                'InputMatrix', inputMatrix, ...
                'DisturbanceMatrix', inputMatrix, ...
                'InputSpace', Polyhedron('lb', 0, 'ub',
 auto_input_max), ...
                'Disturbance', process_disturbance);
disp(sys)
```

```
Linear time invariant system with 3 states, 1 inputs, and 1
 disturbances.
```

Safety specifications

We desire that the state remains inside a set $\mathcal{K} = \{x \in \mathbf{R}^3 : 0 \leq x_1 \leq 6, 0 \leq x_2 \leq 10, 0 \leq x_3 \leq 10\}$.

```
time_horizon = 5;
safe_set = Polyhedron('lb',[1, 0, 0], 'ub', [6, 10, 10]);
safety_tube = Tube('viability',safe_set, time_horizon);
```

Computation of the underapproximation of the stochastic viability set

We are interested in computing the stochastic viability set at probability 0.99.

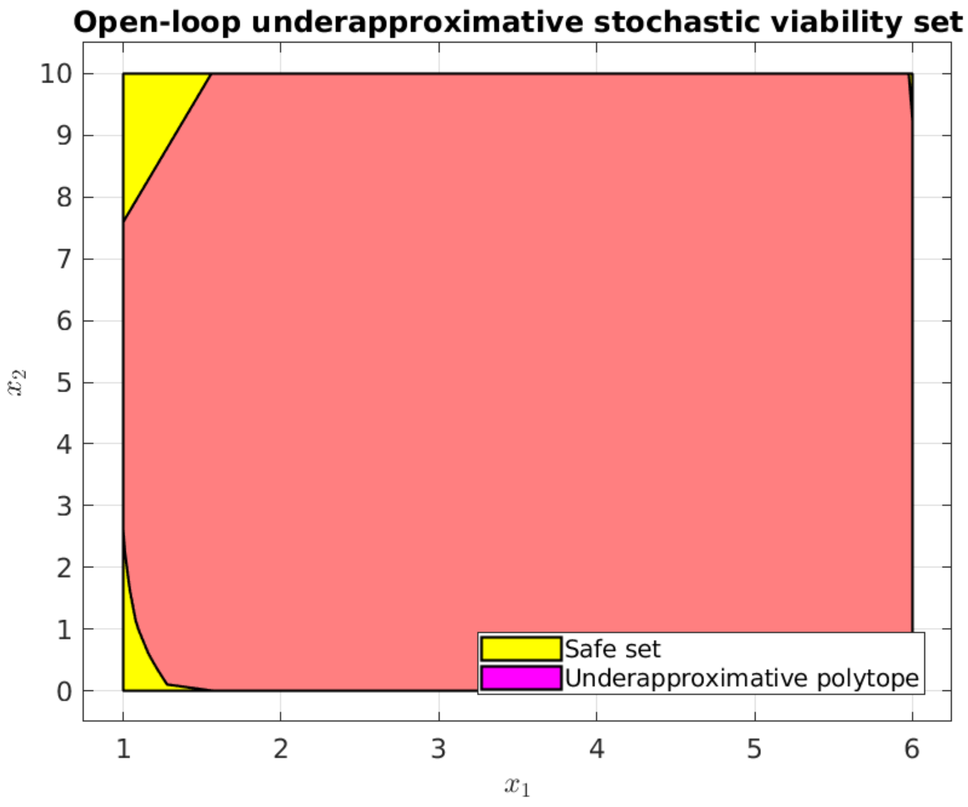
For using `SReachSet` with `chance-open` option, we need a set of direction vectors and an affine hull (n-2 dimensional) intersecting the initial state. Since x_3 of the dynamics is slow, we fix it $x_3 = 5$ and analyze the rest of the system.

```
% Safety probability threshold of interest
% ----------------------------------------
prob_thresh = 0.99;     % Stochastic reach-avoid 'level' of interest
% Definition of the affine hull
% ----------------------------
x3_initial_state = 5;
init_safe_set_affine = Polyhedron('He',[0, 0, 1, x3_initial_state]);
% Definition of set of direction vectors
% -------------------------------------
no_of_dir_vecs = 32;
theta_vec = linspace(0,2*pi, no_of_dir_vecs);
set_of_dir_vecs =
 [cos(theta_vec);sin(theta_vec);zeros(1,no_of_dir_vecs)];
% Use SReachSet to compute the underapproximative set
% ---------------------------------------------------
% Use Ctrl + F1 to get the hints
options = SReachSetOptions('term','chance-open', ...
    'set_of_dir_vecs', set_of_dir_vecs,...
    'init_safe_set_affine', init_safe_set_affine);
timer_val = tic;
[underapprox_stoch_viab_polytope, extra_info] = SReachSet('term',...
    'chance-open', sys, prob_thresh, safety_tube, options);
elapsed_time = toc(timer_val);
disp(elapsed_time)
```

```
   11.9967
```

Plotting the stochastic viable set

```
figure(1);
hold on;
safe_set_2D = safe_set.intersect(init_safe_set_affine);
plot(safe_set_2D, 'color', 'y');
plot(underapprox_stoch_viab_polytope, 'color', 'm', 'alpha', 0.5);
leg=legend({'Safe set','Underapproximative polytope'});
set(leg,'Location','SouthEast');
xlabel('$x_1$','interpreter','latex')
ylabel('$x_2$','interpreter','latex')
box on;
grid on;
view([0,90]);
title('Open-loop underapproximative stochastic viability set');
```



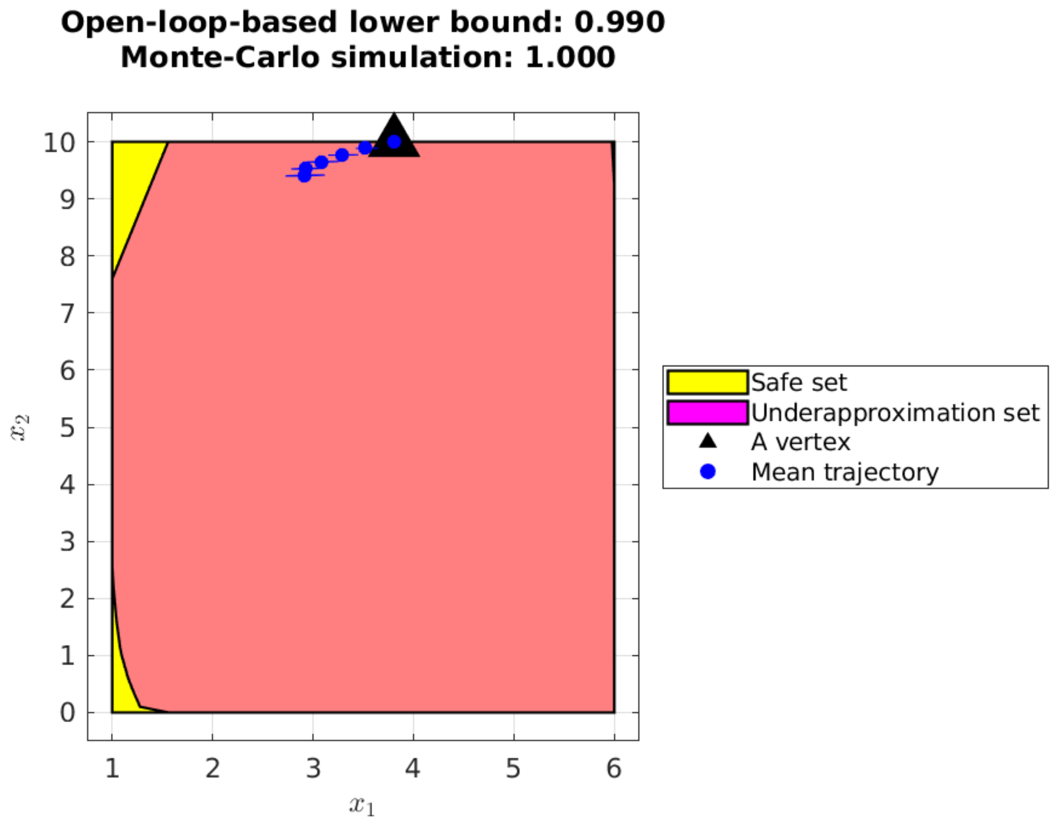
Validate the underapproximative set and the controller using Monte-Carlo

We will now check how the optimal policy computed for each corners perform in Monte-Carlo simulations.

```
n_mcarlo_sims = 1e5;
vertex_indx = 8;
if ~isEmptySet(underapprox_stoch_viab_polytope)
    % Obtain info about the vertices from extra_info struct given by
 SReachSet
    %
 -------------------------------------------------------------------------
    initial_state =
 extra_info(1).vertices_underapprox_polytope(:,vertex_indx);
    opt_input_vec =
 extra_info(1).opt_input_vec_at_vertices(:,vertex_indx);
    stoch_viab_prob_lb = extra_info(1).opt_reach_prob_i(vertex_indx);
    % Compute Monte-Carlo realizations
    % --------------------------------
    concat_state_realization = generateMonteCarloSims(n_mcarlo_sims,
 sys,...
        initial_state, time_horizon, opt_input_vec);
    % Optimal mean trajectory generation
    % ----------------------------------
    [~, H, ~] = sys.getConcatMats(time_horizon);
    sysnoi =
 LtvSystem('StateMatrix',sys.state_mat,'DisturbanceMatrix',...
        sys.dist_mat,'Disturbance',sys.dist);
    [mean_X_sans_input, ~] = SReachFwd('concat-stoch',sysnoi,
 initial_state,...
        time_horizon);
    optimal_mean_X = reshape(mean_X_sans_input + H * opt_input_vec,...
        sys.state_dim,[]);
    % Monte-Carlo estimate of the safety probability
    % ----------------------------------------------
    mcarlo_result = safety_tube.contains(...
        [repmat(initial_state,1,n_mcarlo_sims);
         concat_state_realization]);
    stoch_viab_prob_mc_estim = sum(mcarlo_result)/n_mcarlo_sims;
    fprintf(['Open-loop-based lower bound and Monte-Carlo simulation
 ',...
            '(%1.0e particles): %1.3f, %1.3f\n'],...
            n_mcarlo_sims,...
            stoch_viab_prob_lb,...
            stoch_viab_prob_mc_estim);

    % Plotting
    % --------
    figure(2);
    hold on;
    plot(safe_set_2D.slice(3,x3_initial_state), 'color', 'y');
    plot(underapprox_stoch_viab_polytope.slice(3,x3_initial_state),...
        'color','m','alpha',0.5);
    scatter(initial_state(1),initial_state(2), 300,'k^','filled');
    % Plot the optimal mean trajectory from the vertex under study
    scatter([initial_state(1), optimal_mean_X(1,:)],...
            [initial_state(2), optimal_mean_X(2,:)],...
          30, 'bo', 'filled');
    legend_cell = {'Safe set', 'Underapproximation set', 'A
 vertex',...
        'Mean trajectory'};
    leg = legend(legend_cell,'Location','EastOutside');
    ellipsoidsFromMonteCarloSims(concat_state_realization,
 sys.state_dim,...
        [1,2], {'b'});
    title(sprintf(['Open-loop-based lower bound: %1.3f\n Monte-Carlo
 ',...
                   'simulation: %1.3f\n'], stoch_viab_prob_lb,...
                    stoch_viab_prob_mc_estim));
    box on;
    grid on;
    xlabel('$x_1$','interpreter','latex')
    ylabel('$x_2$','interpreter','latex');
end
```

```
Open-loop-based lower bound and Monte-Carlo simulation (1e+05
 particles): 0.990, 1.000
```



Published with MATLAB® R2017b